A Novel Approach to Low Multiplicative Complexity Logic Design

Jia Jun Tay, Ming Ming Wong
Faculty of Engineering, Computing and Science,
Swinburne University of Technology Sarawak Campus,
Sarawak, Malaysia
e-mail: jtay@swinburne.edu.my

M. L. Dennis Wong
Heriot-Watt University Malaysia,
Putrajaya, Malaysia
e-mail: D.Wong@hw.ac.uk

Cishen Zhang, Ismat Hijazin
Faculty of Science, Engineering and Technology,
Swinburne University of Technology Hawthorn Campus,
Victoria, Australia

***Abstract*—Logic optimization over the logic basis (AND, XOR, NOT) has received increased attention in recent works due to the potential in low gate count logic circuit implementation. Previous logic minimization heuristic in this logic basis involved randomized selection processes and thus exhibits uncontrolled variations in the results produced and algorithm execution time. In this work, we demonstrate a novel approach to the same problem using Positive Polarity Reed-Muller factorization. The proposed algorithm eliminates the reliance on randomness and produces all optimal solutions obtainable through the factorization method. This enables the application of different selection criteria post-optimization to maximize circuit sharing between functions. The proposed algorithm is aimed towards optimizing the S-boxes of lightweight cryptographic schemes.**

Keywords-logic minimization; positive polarity reed-muller; multiplicative complexity; lightweight cryptography

I. INTRODUCTION

Logic design is an exercise of interconnecting basic logic building blocks to perform a specific function. The goal is to design an economical circuit for a desired function which is optimal in regards to a meaningful metric. Depending on the application, the meaningful metric ranges from gate count to circuit depth or even power consumption. Conventional methods of logic simplification, such as Karnaugh mapping and Quine-McCluskey procedure, operate over the logic basis (AND, OR, NOT). However, recent works have begun exploring logic design over the logic basis (AND, XOR, NOT). In general, it has been observed that logic expressions over the logic basis (AND, XOR, NOT) require fewer product terms than their alternative expressions over the logic basis (AND, XOR, NOT) [1]. This property gives the potential for lower gate count logic circuits which are crucial for resource constrained applications. As such, the potential of an efficient logic minimization algorithm may aid the purpose of optimizing the Substitution boxes (S-boxes) of popular lightweight cryptographic schemes.

Works in [2] observed that a logic circuit constructed over the logic basis (AND, XOR, NOT) using the minimal number of AND gates usually results in very low gate count. This observation inspired a heuristic approach to logic minimization patented in [3]. Although the algorithm has been demonstrated to produce good results on popular cryptographic S-boxes such as the $GF(2^4)$ AES S-box [4] and the PRESENT S-box [5], it experiences consistency issues due to the nature of the randomized selection process involved. Even with improvements proposed in [6], we are unable to completely eliminate the relevance of randomness in the algorithm.

In this work, we propose a novel approach to solving low multiplicative complexity logic minimization problem through Positive Polarity Reed-Muller (PPRM) factorization. This method is aimed towards solving logic design problems for S-boxes of lightweight cryptographic schemes. Therefore, we limit our discussion to logic problems with 4-bit inputs as most popular lightweight cryptographic schemes utilize one or more 4x4-bit S-boxes for their non-linear substitution processes.

The structure of the paper is organized as follows: Section II and III provide preliminary knowledge regarding the PPRM expressions and the concept of multiplicative complexity in this context. In Section IV, we demonstrate the role of PPRM factorization in achieving optimal multiplicative complexity in logic design. The proposed logic minimization algorithm is described in Section V. In Section VI, we present optimal expressions for each individual function in the PRESENT S-box. Section VII explains the benefits of the proposed algorithm in relation to the previous algorithm. Finally, concluding remarks and potential future works are discussed in Section VIII.

II. POSITIVE POLARITY REED-MULLER EXPRESSIONS

The first step to solving any logic design problem is to derive a logic equation which describes the circuit outputs as a function of the circuit inputs. Similar to expressing a function in its sum-of-product (SOP) form in the logic basis (AND, OR, NOT), a PPRM expression is the function's equivalent in the logic basis (AND, XOR, NOT). The term "Positive Polarity" implies that all literals in the Reed-Muller

expression appear uncomplemented. In general, given a function with n -bit inputs $x_1, x_2, \cdots, x_n$, its PPRM expressions can be defined as shown in (1) where $a_0, a_1, \cdots, a_{2^n-1} \in \{0,1\}$ and $\pi_0, \pi_1, \pi_2, \pi_3, \pi_4, \cdots, \pi_{2^n-1} = 1, x_1, x_2, x_2x_1, x_3, \cdots, x_nx_{n-1}\cdots x_1$ fully describe f .

$$f(x_1, x_2, \cdots, x_n) = a_0\pi_0 + a_1\pi_1 + \cdots + a_{2^n-1}\pi_{2^n-1} \quad (1)$$

One important property of the PPRM expression is that for each unique function, there is only one unique representation in PPRM form. This unique PPRM expression can be derived easily by multiplying the truth vector of the function with the n -variable transform matrix T_n given in (2) where $T_0 = [1]$.

$$T_n = \begin{bmatrix} T_{n-1} & 0 \\ T_{n-1} & T_{n-1} \end{bmatrix} \quad (2)$$

Logical and mathematical notations in PPRM expressions are similar to the operations in modulo-2 arithmetic. In this work, we express logic XOR (mod-2 addition) using notations shown in (3). In this case, c represents the *sum* resulting from the addition of a and b .

$$a \oplus b = a + b = c \quad (3)$$

Similarly, logic AND (mod-2 multiplication) is expressed using notations shown in (4), where c is the *product* resulting from the multiplication of a and b .

$$a \bullet b = ab = c \quad (4)$$

III. Multiplicative Complexity

Multiplicative complexity $c_\wedge(f)$ of a function is defined as the minimal number of multiplication (AND gates) required to realize the function over the basis (AND, XOR, NOT). It has been demonstrated in [7] that for a function of degree d , the multiplicative complexity is at least $d-1$, i.e. $c_\wedge(f) \geq d-1$. Although a definite upper bound for the multiplicative complexity of a function has yet to be established, [8] has proven that for a function with n inputs, its multiplicative complexity is at most $n-1$ as long as $n \leq 5$, i.e. $c_\wedge(f) \leq n-1$ given $n \leq 5$.

Low multiplicative complexity logic minimization aims to achieve low gate count circuits for a desired function by using the minimal number of AND gates required to construct the circuit. As such, the proposed algorithm can be described as a logic minimization heuristic that produces logic circuits which are optimal in terms of multiplicative complexity.

IV. Positive Polarity Reed-Muller Factorization for Low Multiplicative Complexity

Previous approach to low multiplicative complexity logic minimization requires performing addition and multiplication on random selection of signals within an expanding sample space until the desired truth vector is found. In this section, we demonstrate a different approach to achieving minimal AND gates through PPRM factorization. As mentioned previously, we consider four inputs logic functions for application in lightweight cryptographic S-boxes.

Let $x_1, x_2, \cdots, x_4$ be the four logic inputs of a function. Consider the PPRM expression given in (5).

$$f(x) = x_1x_2x_3 + x_1x_2 + x_1x_4 \quad (5)$$

Using the lower bound rule, it is easy to see that the function has a multiplicative complexity of $c_\wedge(f) = d-1 = 2$. In this case, it is obvious that we can achieve optimal AND gates by factorizing the literal x_1 as shown in (6).

$$f(x) = x_1(x_2x_3 + x_2 + x_4) \quad (6)$$

However, actual logic problems are often not solvable by a single-staged factorization. Consider the PPRM expression in (7), which describes the function for the most significant output bit of the PRESENT S-box [9].

$$f(x) = x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3 + x_1 + x_2 + x_4 + 1 \quad (7)$$

The multiplicative complexity of the function is known to be $c_\wedge(f) = 2$. However, there is no obvious way to manipulate the PPRM expression into a form that requires two AND gates to realize. To solve the problem, we first remove any literals with a degree of $d \leq 1$ to form $f'(x)$. These literals are completely linear (no multiplication) and thus can be reintroduced into the equation through additions once $f'(x)$ is optimized.

$$f'(x) = x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3 \quad (8)$$

From (8), we proceed with the first step of factorization. There are no specific rules in choosing the literal to be factorized at this stage as each may potentially lead to an optimal solution. Factorizing x_2 will result in the expression in (9).

$$f'(x) = \underbrace{x_2}_{a}\underbrace{(x_1x_3 + x_1x_4 + x_3)}_{b} + \underbrace{x_1x_3x_4}_{c} \quad (9)$$

Notice the three important sections for the resultant expression as indicated in (9). a includes the factorized literal, b includes the factorized clauses, while c represents the remaining clauses that do not contain the factorized literal. At this stage, if ab can be achieved with AND gates less than or equal to $c_\wedge(f) = 2$, there is a possibility that an optimal solution can be derived from the expression. As shown in (10), ab can be realized with two AND gates in this case.

$$ab = x_2(x_1(x_3 + x_4) + x_3) \quad (10)$$

The next objective is to manipulate the expression in such as way that c requires no extra AND gates to realize. In other words, c needs to become either (a) free of multiplication or (b) has the same clauses as b . This can be done by adding another literal into a . For example, by

adding x_1 into a, a new string of clauses d has to be added to c for cancellation. This is shown in (11).

$$f'(x)=\underbrace{(x_2+x_1)}_{a}\underbrace{(x_1x_3+x_1x_4+x_3)}_{b} +\underbrace{x_1x_3x_4}_{c}+\underbrace{x_1x_3+x_1x_4+x_1x_3}_{d} \quad (11)$$

Completing the XOR operations between c and d results in (12).

$$f'(x)=\underbrace{(x_2+x_1)}_{a}\underbrace{(x_1x_3+x_1x_4+x_3)}_{b}+\underbrace{x_1x_3x_4+x_1x_4}_{c} \quad (12)$$

With the expression manipulation method above, we are able to change the clauses in c without introducing additional AND gates to ab. Since c in (12) still requires additional AND gates after ab, by adding the literal x_3 into a and completing the addition between c and d, we arrive at the expression in (13).

$$f'(x)=\underbrace{(x_2+x_1+x_3)}_{a}\underbrace{(x_1x_3+x_1x_4+x_3)}_{b} +\underbrace{x_1x_3+x_1x_4+x_3}_{c} \quad (13)$$

$$f'(x)=\underbrace{(x_2+x_1+x_3)}_{a}\underbrace{(x_1(x_3+x_4)+x_3)}_{b} +\underbrace{x_1(x_3+x_4)+x_3}_{c} \quad (14)$$

Since the clauses in c is exactly identical to the clauses in b, no extra AND gates are required and the entire expression in (14) can be realized with two AND gates, i.e. optimal in terms of multiplicative complexity. The final step is to reintroduce the linear section of the circuit removed in the first step to obtain the optimized expression (15).

$$f(x)=(x_2+x_1+x_3)(x_1(x_3+x_4)+x_3) +x_1(x_3+x_4)+x_3+x_1+x_2+x_4+1 \quad (15)$$

This example demonstrates the procedure to derive an optimal solution for a low multiplicative complexity logic design problem without relying on randomness. It is important to note that although the length of (15) may suggest a large logic circuit, a huge portion of the equation are linear strings of XORs which share repeated terms. As such, the actual logic circuit required to realize the expression is much smaller due to circuit sharing. The challenge now is to derive an efficient algorithm that produces all optimal solutions achievable through this method for a given problem.

V. Proposed Algorithm for Low Multiplicative Complexity Logic Minimization

In this section, we describe the proposed algorithm designed using concept demonstrated in Section IV. A pseudocode for the algorithm is provided in Algorithm 1.

TABLE I. Frequencies of Literals in Each Degree for (8)

Literal	Degree				Total
	4	3	2	1	
x_1	0	3	0	0	3
x_2	0	2	1	0	3
x_3	0	2	1	0	3
x_4	0	2	0	0	2

A. Step 1: Select the First Literal for Factorization

The algorithm selects a literal to be factorized from the available n inputs. The selection is done sequentially as all literals need to be factorized eventually to discover all possible solutions unless they are interchangeable (see Subsection V-F).

B. Step 2: Form a, b and c

Once the literal to be factorized is chosen, the algorithm forms a, b and c as demonstrated in (9). Note that after the factorization of the first literal, b will remain constant throughout the subsequent steps of the algorithm.

C. Step 3: Form All Possible Variations of d_i

Excluding the factorized literal, there are $n-1$ possible literals to be added into a. As such, there are $n-1$ possible variations of d_i to be added into c. The algorithm forms all variations in a set D, in which $d_1, d_2, \cdots, d_{n-1} \in D$.

D. Step 4: Form All Possible Combinations of d_i

There are 2^{n-1} possible combinations of elements in set D through addition (including 0), i.e. $0, d_1, d_2, d_1+d_2, \cdots, d_1+d_2+\cdots+d_{n-1}$. The algorithm forms all possible combinations in a new set D', in which $0, d_1, d_2, d_1+d_2, \cdots, d_1+d_2+\cdots+d_{n-1} \in D'$.

E. Step 5: Check If Any Elements of D' Solve c

The algorithm then adds each of the 2^{n-1} elements in D' into c and determine if (a) the sum is free of $d>1$ clauses or (b) the sum can be implemented without additional AND gates after ab. Elements of D' that satisfy (a) or (b) will give an optimal solution for the problem.

F. Step 6: Return to Step 1 for the Next Literal

For a given problem, the algorithm needs to attempt factorization of all literals to discover all possible solutions. As such, the algorithm needs to perform multiple loops of the previous steps equal to the number of inputs n. However, if two or more literals have the same frequencies in every degree of the clauses (in the PPRM expression), they are mutually interchangeable and the algorithm need only attempt factorization for one the interchangeable literals.
For example, Table I gives the frequency spread of literals for (8). Notice that x_2 and x_3 have the identical frequency spread and are thus mutually interchangeable. Since factorizing x_2 produced the result in (14), by nature of the

interchangeable property, we can replace all instances of x_2 with x_3 in (14) and vice versa to obtain an alternative optimal solution (16).

$$f'(x) = (x_3 + x_1 + x_2)(x_1(x_2 + x_4) + x_2) + x_1(x_2 + x_4) + x_2 \quad (16)$$

As such, when interchangeable literals are present, the algorithm only needs to repeat the factorization process for the non-interchangeable literals.

Algorithm 1 Pseudocode for proposed algorithm

1: Initialization
2: $i = 1$
3: Select literal x_i
4: Form a, b and c through factorization
5: Form $d_1, d_2, ..., d_{n-1} \in D$
6: Form $0, d_1, d_2, d_1 + d_2, ..., d_1 + d_2 + ... + d_{n-1} \in D'$
7: $p = 1$
8: $s = D'_p + c$
9: If s does not require multiplication, proceed to line 12
10: Else if clauses in s = clauses in b, proceed to line 12
11: Else proceed to line 13
12: Record optimal solution in set S
13: $p = p + 1$
14: If $p \leq size(D')$, return to line 8
15: If $i = n$, proceed to line 19
16: Else $i = i + 1$
17: If x_i is interchangeable with previous literal, return to 16
18: Else return to line 3
19: End

TABLE II. OPTIMAL EXPRESSIONS FOR PRESENT S-BOX FUNCTIONS

Function	$c_\wedge(f)$	Optimal expression
f_1	1	$x_2x_3 + x_1 + x_3 + x_4$
f_2	2	$(x_2 + x_1 + x_3)(x_1(x_3 + x_4) + x_4) + x_2 + x_4$
		$(x_3 + x_1 + x_2)(x_1(x_2 + x_4) + x_4) + x_2 + x_4$
f_3	2	$(x_1 + x_4)(x_4(x_2 + x_3) + x_2 + x_1) + x_4(x_2 + x_3) + x_3 + 1$
		$(x_4 + x_1)(x_1(x_2 + x_3) + x_2 + x_1) + x_1(x_2 + x_3) + x_1 + x_3 + x_4 + 1$
f_4	2	$(x_2 + x_1 + x_3)(x_1(x_3 + x_4) + x_3) + x_1(x_3 + x_4) + x_3 + x_1 + x_2 + x_4 + 1$
		$(x_3 + x_1 + x_2)(x_1(x_2 + x_4) + x_2) + x_1(x_2 + x_4) + x_1 + x_4 + 1$

VI. RESULTS ON PRESENT S-BOX

We performed optimization on each individual function in the PRESENT S-box [9] independently using the proposed algorithm. f_1 represents the least significant output bit of the S-box while f_4 describes the most significant bit. This demonstration is not meant to produce the smallest PRESENT S-box for comparison but rather to verify the ability of the proposed algorithm to find the optimal expressions for each individual functions.

TABLE III. COMPARISON OF EXECUTION TIME BETWEEN ORIGINAL AND PROPOSED ALGORITHMS WHEN SOLVING (7)

Algorithm	Time (s)
Original [6]	7.3758
Proposed	0.0081

All optimal expressions produced in Table II can be realized with the minimum number of AND gates as given by their respective $c_\wedge(f)$ values, i.e. they are optimal in terms of multiplicative complexity. f_2, f_3 and f_4 each have two possible optimal solutions. Therefore, additional selection criteria can be implemented post-optimization to identify the ideal solution for each function which enables the greatest degree of circuit sharing between the four functions for the construction of the full S-box.

VII. ADVANTAGES OF PROPOSED ALGORITHM

In this section, we highlight the advantages of the proposed algorithm in cryptographic or similar applications. They are mostly done in comparison to the original algorithm described in [3] and the improved version in [6].

A. *Consistent Results*

As mentioned previously, the main motivation to derive a new heuristic algorithm to solve the low multiplicative complexity logic design problem is to attempt to eliminate the need for randomized selection present in the previous algorithm. In this regard, the proposed algorithm succeeds in producing consistent results in every execution. As opposed to the previous algorithm where each execution presents a random optimal result, the proposed algorithm produces all possible optimal results achievable through PPRM factorization.

B. *Enables Selection Criteria for Multiple-Output Problems*

In this work, we have limited our discussion to single-output problem (one function) to demonstrate the PPRM factorization process in detail. However, most actual applications such as cryptographic S-boxes have multiple outputs expressed as functions of the same set of inputs. An optimal solution for a single function may not be optimal when considered as part of the multiple-output problem. Since the proposed algorithm produces all possible optimal solutions achievable through PPRM factorization, selection criteria can be introduced to select the best solutions once each function has been optimized. In general, selection criteria that enables the most circuit sharing between the multiple functions are the most desirable.

C. *Faster and Consistent Execution Time*

Due to the randomness involved in the original algorithm, comparing the algorithmic complexity of both algorithms is difficult. To make a comparison on the execution time, both algorithms are applied to optimize function (7) using MATLAB R2012b. The original algorithm is sampled for 100 trials to obtain the average value. The execution time for the proposed algorithm showed negligible variances between samples.

In Table III, the proposed algorithm showed a significant improvement in terms of execution time compared to the original algorithm. Although execution time varies with the function under optimization, we observed similar degree of difference when comparing both algorithms using random functions. Also, it is important to note that the proposed algorithm produced two optimal solutions within the duration of the execution time (see f_4 in Table II) while the original algorithm produced one random optimal result.

VIII. Conclusion and Future Work

We present a novel approach to low multiplicative complexity logic minimization based on the concept of PPRM factorization. The proposed algorithm enabled logic minimization over the logic basis (AND, XOR, NOT) to be achieved without reliance on randomness. Multiple optimal solutions produced by the algorithm allow the use of selection criteria when solving for multiple-output problems. This enables the potential for a greater degree of circuit sharing when optimizing logic circuits for lightweight cryptography. The proposed algorithm also showed significant improvement in execution time when compared to the original algorithm under the same logic problem.

The main motivation of this work is to derive an efficient logic minimization algorithm for applications in lightweight cryptography. As such, the current algorithm is insufficient as it is only capable of optimally solving single-output problems. However, by experimenting with different selection criteria to promote circuit sharing between multiple functions that share the same input set, we hope to expand the algorithm into producing economical logic circuits that are optimal in terms of multiplicative complexity for multiple-output problems. Similarly, we see potential in expanding the algorithm to solve logic problems with higher number of inputs using the same concept.

Acknowledgment

This work has been supported in part of the Melbourne-Sarawak Research Collaboration Scheme.

References

[1] T. Sasao and P. Besslich, "On the complexity of mod-2l sum PLA's," *IEEE Trans. Comput.*, vol. 39, no. 2, pp. 262-266, Feb. 1990.

[2] J. Boyar, P. Matthews, and R. Peralta, "Logic Minimization Techniques with Applications to Cryptology," *Journal of Cryptology*, vol. 26, no. 2, pp. 280-312, Apr. 2013.

[3] R. C. Peralta and J. Boyar, "Method of optimizing combinational circuits," U.S. Patent 8 316 338 B2, Nov. 20, 2012.

[4] J. Boyar and R. Peralta, "A Small Depth-16 Circuit for the AES S-Box," in *IFIP International Information Security Conference, SEC 2012: Information Security and Privacy Research*, Heraklion, Crete, Greece, 2012, pp. 287-298.

[5] N. T. Courtois, D. Hulme, and T. Mourouzis, "Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis," in *IACR Cryptology ePrint Archive: Report 2011/475*, 2011.

[6] J. J. Tay, M. L. D. Wong, M. M. Wong, C. Zhang, and I. Hijazin, "Low multiplicative complexity logic minimisation over the basis (AND, XOR, NOT)," *Electronics Letters*, vol. 52, no. 17, pp. 1438-1440, Aug. 2016.

[7] C. P. Schnorr, "The multiplicative complexity of boolean functions," *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, vol. 357, pp. 45-58, July 1988.

[8] M. S. Turan and R. Peralta, "The Multiplicative Complexity of Boolean Functions on Four and Five Variables," in *Lightweight Cryptography for Security and Privacy: Third International Workshop, LightSec 2014*, Istanbul, Turkey, 2014, pp. 21-33.

[9] A. Bogdanov *et al.*, "PRESENT: An Ultra-Lightweight Block Cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop*, Vienna, Austria, 2007, pp. 450-466.